6. Francisco

Program Name: Francisco.java **Input File: francisco.dat**

Francisco loves to play the game, **Tile Bonanza**, which uses a square board with black and white tiles. The goal of this game is to rearrange the pieces into an optimal configuration with a limited number of moves. The rules for a move are that any one piece can be moved to any empty space among its adjacent locations (up, down, left, right, or any of the four diagonal locations). A piece can only be moved to an empty spot. The optimal configuration encourages grouping of pieces of like color and separating pieces of different color. To calculate the scoring, for every neighboring piece of like color in an adjacent square, 10 points is awarded. For each adjacent piece of opposite color, 5 points is deducted. Given this scoring system, with the number of moves indicated, find the optimal configuration and report the score, and output the board as well if only one optimal configuration is possible.

```
B 0 0
0 W 0
W 0 B
```

For example, in the board to the left, before any moves are made, the score is 0. Francisco can see this by looking at each piece on the board individually. Going from left to right, top to bottom, the first Black piece has a White piece in one of its adjacent squares, bringing the total to -5. The White piece in the center has a White and two Blacks next door, which totals 0, keeping the score at -5. The next White piece in the bottom left has an adjacent White piece, which adds 10, bringing the total to 5. Finally, the Black piece in the bottom right has an adjacent White piece, subtracting 5, bringing the total score to 0.

Given exactly one allowed move, the optimal configurations to maximize the score for this board results in the following two optimal scoring boards, both with a point total of 10. The middle White piece can either move down one space, or left one space, both moves resulting in a score of 10.

```
B 0 0
0 0 0
W W B

B 0 0
W 0 0
W 0 B
```

Given a board where the digit 0 represents a space, the digit 1 represents Black and the digit 2 represents White, and the number of allowed moves to find an optimal score, print that score and the number of boards that result in that score. If there is only one configuration that yields the optimal score, display that configuration.

Input: The first integer will represent the number of data sets to follow. The first integer M (0<M<=K) of each data set represents the number of allowed moves. The next integer, K (2<K<=10), will be the size of the board. The next K lines contain a square grid made up of the digits 0, 1, and 2 (single spaces separating each digit, no blank lines between rows) representing the initial configuration of the board. There will always be the same number of black and white pieces, and at least one piece of each color. There will always be at least one empty square.

Output: The maximum score is prefixed by the sentence: "**`MAXIMUM SCORE: <score>`** ", followed by "**`THERE ARE <num> OPTIMAL BOARDS.`**" where **`<score>`** represents the optimal score, and **`<num>`** represents the number of different configurations that yield that maximum score. If there is only one such board, print "**`THERE IS 1 OPTIMAL BOARD.`**", and then also **print the board immediately below**. Print a blank line after each output set.

Sample Input:

```
3
1
3
1 0 0
0 2 0
2 0 1
1
3
0 0 1
0 2 1
2 0 0
3
5
0 0 1 0 2
0 2 0 2 0
1 0 1 0 1
0 2 0 2 0
0 0 1 0 0
```

Sample Output:

```
MAXIMUM SCORE: 10
THERE ARE 2 OPTIMAL BOARDS.

MAXIMUM SCORE: 40
THERE IS 1 OPTIMAL BOARD.
0 0 1
2 0 1
2 0 0

MAXIMUM SCORE: 70
THERE ARE 8 OPTIMAL BOARDS.
```